What is Dev-ops?

Dev-ops is practice/culture that we follow to improve the ability to deliver for organization. (End goal)

Focus on increasing efficiency by automation.

Roadmap 2024:

1. What exactly is dev-ops?
2. Why an organization need to implement dev-ops?
3. Advantage to Org. with SDLC

Always get/ask what and why aspect of any tool / tech that will give us fundamentals.

**Basics and must learn.**

* Learn Linux
  + How Linux is different from window/mac?
  + Why org. use Linux for servers and security?
  + How to create a Linux machine? (Refer AWS)
  + How to create Linux machine on AWS
  + Understand learn basic Linux / shell command. (eg: checking RAM, creating file etc)
* Learn concept of virtualization
  + What exactly is virtualization?
  + What bring people to use virtualization?
  + How virtualization Impact on organization and software industry?
* Learn Shell scripting.
  + Example, write shell scripting for getting info. Of virtual machine.
  + Learn basic and intermediate scripting.
* Learn GIT (I know basics)
  + Version control system
  + Learn git command.
* Learn concept of CI/CD (normal idea as per PO project)
  + It’s not straight forward.
  + Spend a week to understand CI/CD.
  + Start with Jenkins / GitHub actions.
  + For now, focus more on **CI**
  + Cover CD later (reason: In involves Kubernetes and dockers)
  + How to connect Jenkins with Git and perform build operation using build languages such as Maven.
  + Deployment on EC2 instance
* Learn Configuration management.
  + What is configuration management?
  + Why configuration management?
  + Learn Ansible (playbooks)
    - Connect to 2 VM using ansible and install httpd in them.
* Infrastructure Automation
  + What and why we need Infrastructure Automation?
  + What and why Terraform for infra-automation?
  + Write terraform scripts to create EC2 instance on AWS.
* Learn containerization.
  + What is containerization?
  + Why we need containerization?
  + How VM are different from container?
  + Why and how containers are better than VMs?
  + Focus on Docker (it’s a containerization tool)
    - Understand concept of Docker
    - How we build container using Dockers?
    - How we ship the container using Docker?
    - What is volume in docker?
    - What is networking in docker?
    - How to reduce image size using docker?
* Learn Kubernetes
  + What is Kubernetes?
  + Why we need Kubernetes?
  + Understand concept of Kubernetes?
  + How Docker as containerization is different from Kubernetes platform?
  + Difference between container and pod?
  + What is deployment in Kubernetes?
  + What are services in Kubernetes?

Playlist: <https://www.youtube.com/watch?v=Ou9j73aWgyE&list=PLdpzxOOAlwvIKMhk8WhzN1pYoJ1YU8Csa&pp=iAQB>

**Advance Dev-Ops**

* Build CI/CD
* Complex shell scripting
* Learn about monitoring.
* Do projects on Kubernetes.

**Learn Cloud platform**

* Learn AWS as priority (2nd Azure)
* AWS has 200 services
* <https://youtube.com/playlist?list=PLdpzxOOAlwvLNOxX0RfndiYSt1Le9azze&si=0MfARWC3lG4O-d0r>

**Learn Cloud Terraform advance**

* <https://youtube.com/playlist?list=PLdpzxOOAlwvI0O4PeKVV1-yJoX2AqIWuf&si=IezWnaBqi9IYfW4r>

**Learn Python**

Learn python for API and serverless scripting

<https://youtube.com/playlist?list=PLdpzxOOAlwvKwTyYNJCUwGPvql0TrsPgv&si=veX5YgWUK_mL3HRL>

**Order**

Dev-ops 🡪AWS 🡪 Terraform 🡪 Python

**Day 1 – Fundamental of Dev-Ops**

1. **What is Dev-Ops? Or Define Dev-Ops?**

Generally, the definition of Dev-ops changes org., to org. but basically the Dev-ops is **practice / Culture** that an organization / company adopts to **improve their ability to deliver** products (i.e: Application, goals etc.). Where the development and operation go together to work efficiently.

Dev-Ops is about Improving and increasing your ability to deliver an application IN QUICK manner by insuring.

1. Proper **Automation** in place (CI/CD)
2. **Quality** (Quality of code, application / Quality standard is maintained)
3. Continuous **Monitoring** (Issue reporting)
4. Continuous **Testing**

End of the day, there should be no manual process, or one should fasten the existing application process.

Eg: Reducing 4 weeks of process to 2 weeks by insuring above are achieved properly.

1. **Why Dev-Ops?**

**History**

10 years ago, or so when Dev-ops engg. Or Practice was not available, the delivery of product/ application was long manual process.

Eg:

**Developer** used to write code and push to centralized code repo or version control system from personal system. Then the **system admin** used to create server (VM ware etc). Later the **Server Admin** used to create application server (app server) on Server created by System admin engg. Later the application was dealt by **BRE (Build and release engineer),** where BRE deploy application to app server (staging server / prod server), which was later tested by **tester** and then BRE present it to **Customer**.

Developer 🡪 System admin engg 🡪 Server admin engg 🡪 BRE 🡪Tester🡪Customer

Above was long manual process. Entire communication or system was not effective and collaborative due to multiple team operating with different.

**Today:**

* We have collaborative single team unlike 10 years ago.
* One single team following a culture / way of work.
* Focuses on new approach (new tech / applications) for increasing efficiency of product.

1. **How to introduce as Dev-Ops Engg?**

This should tend to explain basic intro

**DevOps:**

I am working as Dev-ops engineer, I have past 2 years of experience in dev-ops, before to that I used to work as build and release engineer together with Forgerock IDAM domain.

**Security:**

I am working as IDAM engineer, I have past 2 years of experience in IDAM domain specifically to implementation and configuration in FR AM, before to that I used to work as Java reverse engineer and build and release engineer with Automation in-short as **seasoned Dev-Ops** role in team.

**Why shift from Security to Dev-Ops?**

In Last project I worked on automation of IDAM application, where based on requirement and their manual implementation and successful result, we used to do same using automation pipeline in place.

1. SSOADM application to automate OpenAM.
2. Changes in ansible scripts to deliver values from property file repo to Application-repo.
3. Managed the Repo using git bash.
4. Created version of updated repo code.
5. Deployed application via Jenkins.
6. Maintained the AWS EC2 instances and application (route53, LB, TG, instance etc)
7. Monitored incident on Kibana, AWS cloudwatch and FR application server logs.

**Day 2: Improve SDLC with DevOps**

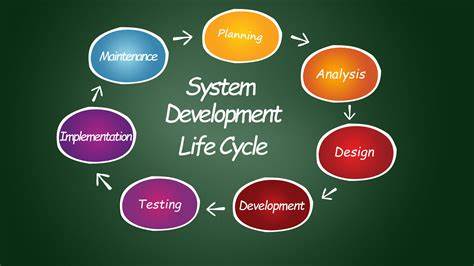
**SDLC:** Software Development Life Cycle

SDLC is process / set of standards followed by an industry / organization to Design, develop, test, and deliver a product application.

End goal of SDLC is to deliver high quality product.

Pillars of SDLC:

1. Planning: Gathering requirements and analyzing then (done by BA and core team)
2. Defining: Define requirements and document them.
3. Designing: HLD and LLD set by Architect and core team developer respectively.
4. Building: Development (front and backend development)
5. Testing
6. Deployment



**Dev-Ops in SDLC**

**Dev-Ops** engg. Fastens these 3 aspects in automated way.

Dev-Ops centric: Building, Testing and Deployment from SDLC

It automates and improve efficiency.

**Day 3: Virtual Machine**

What is a Server?

What is a VM?

What is a Hypervisor?

Difference between Physical and Virtual Machine?

Advantages of Virtual Machines

**Server** is a device / allocated space which It’s designed to manages network resources and provide services to another computer program to host application.

**Hypervisor** is software that helps installing virtual machines (VMs) on bare matal / physical server. In short, it

A hypervisor is a software that you can use to run multiple virtual machines on a single physical machine.

1. Create / Install VM on physical servers.
2. Hypervisors provide logically isolated / separated.

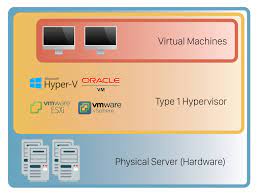
**Popular Hypervisor:**

1. VMware
2. Xen
3. Oracle

**Virtual Machine** are virtual environments which function as virtual computer systems.

Physical Servers 🡪 execute hypervisor on server 🡪 hypervisor creates VM {VM1, VM2, …. VMn}

Each VM {VM1, VM2, …. VMn} is independent of each other having separate CPU, RAM, memory, hardware etc. Hence logically isolated.



**Day 4: Virtual Machine**

**Cloud Provider:** AWS, Azure, GCP etc.

* VM is called EC2 instance in AWS.
* VM is called as virtual machine in Azure.
* Cloud provides infrastructure (VM and supporting service using API)
* We can create an instance using AWS console.
* Creation of multiple VM using UI individually is hectic, there comes Dev-Ops to make it automate for more efficiency.

**Popular tool for Infrastructure automation scripts in AWS:**

1. AWS CLI
2. Aws API
3. AWS CFT
4. AWS CDK
5. Terraform

* AWS CDK is opted over Terraform in case where AWS is dedicatedly used for long term in service.
* Terraform are opted for Hybrid cloud pattern. That is using multiple cloud provider services.

**Automate pipeline for creating EC2 instance.**

Scripts 🡪 AWS EC2 server API + AWS EC2 server 🡪 Exchange of request and response 🡪 it launch instance (VM) with IP and specification.

Task: Create an AWS instance (VM) using AWS cloud services?

Solu: <https://youtu.be/MkIRh1mi8Ms?si=E27wS-1j3Q9OLAQw>

Get the instance IDs IP address.

Instances (VM) can be operated by AWS console, putty (using IP and ppk key or openssh and .pem file)

Public IP: Accessed via Putty / openssh

Private IP: Accessed via Putty / openssh under AWS or VPN

eg: Putty, Mobaxterm, nomachine, MremoteNG

**Day 5:** AWS CLI Full Guide | How to connect to EC2 Instance from UI & Terminal | AWS CFT walk though.

**Automate using AWS CLI**

Download AWS CLI to your OS

* AWS configure
* Provide security credential
* AWS CLI can be used on server to handle AWS cloud operation.

To access VM instance using on ubuntu linux we need ssh pem for access

* ssh ubuntu@<ip>
* ssh -i <location of .pem> ubuntu@<ip>

**How to work with CloudFormation templates(CFT)?**

CFT is alternative of talking to AWS cloud APIs.

AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS.

You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and CloudFormation takes care of provisioning and configuring those resources for you.

You don't need to individually create and configure AWS resources and figure out what's dependent on what; CloudFormation handles that.

We can create stack with CFT to automate delivery.

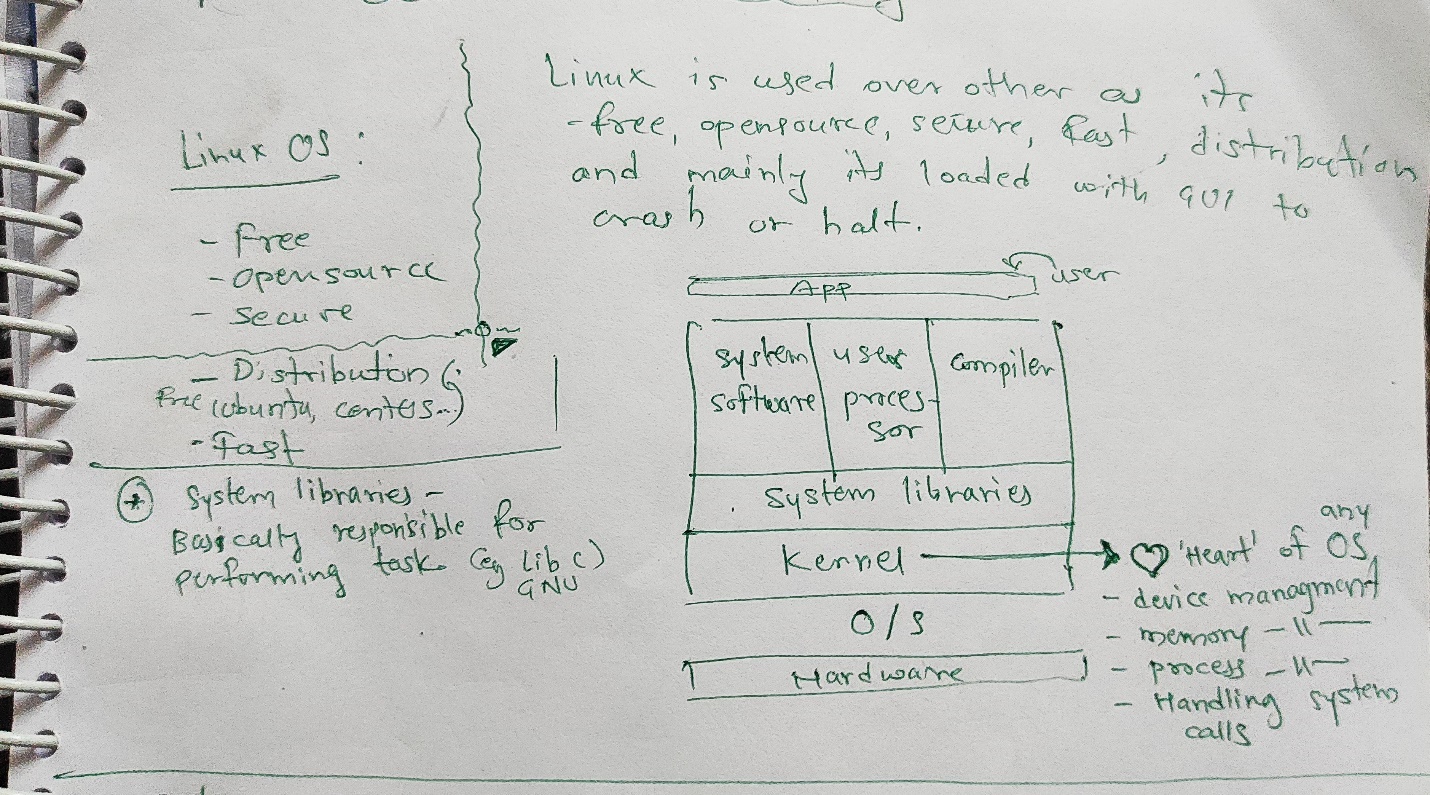
**Git: awslabs/aws-cloudformation-templates**

**Similarly,** Python uses Boto3 package for handling cloud operation like AWS CLI and CFT for AWS or terraform for hybrid cloud operation.

**Day-6 | Linux & Shell Scripting | Complete Shell Scripting Playlist**

­Linux is an operating system (OS), free, open source, secure and distributed over ubuntu, centOS, fast etc..

Linux is used over other OS it’s lightweight and mainly it’s not loaded with GUI, which help server not crash or halt.



Shell scripting is a way (command-line) you talk to your OS or distribution.

